Computer Science 271
Project 0101
Due Monday, March 27

You will complete this project with a partner. Both individuals are expected to contribute equally to all parts of the project, including the proofs.

1. Read Section B.5 (in Appendix B). Then prove the following by induction.

   (a) A complete binary tree with height h contains $2^{h+1} - 1$ total nodes.

   (b) A complete binary tree with n nodes has $(n-1)/2$ internal nodes.

2. Consider a binary search tree T whose keys are distinct. Prove that if the right subtree of a node x in T is empty and x has a successor y, then y is the lowest ancestor of x whose left child is also an ancestor of x. (Recall that every node is its own ancestor.)

3. Implement a binary search tree as a template class. Your binary search tree must support the following operations, in addition to a copy constructor, destructor, and assignment operator.

```
bool empty();                              // return true if empty; false o/w
KeyType *get(const KeyType& k);            // return first element with key equal to k
void insert(KeyType *k);                   // insert k into the tree
void remove(const KeyType& k);             // delete first element with key equal to k
KeyType *maximum();                        // return the maximum element
KeyType *minimum();                        // return the minimum element
KeyType *successor(const KeyType& k);      // return the successor of k
KeyType *predecessor(const KeyType& k); // return the predecessor of k
std::string inOrder();          // return string of elements from an inorder traversal
std::string preOrder();         // return string of elements from a preorder traversal
std::string postOrder();        // return string of elements from a postorder traversal
```

   Notes:

   - Notice that the `insert` and `get` methods insert and return *pointers* to `KeyType` objects. This means that each node needs to contain a *pointer* to an item.
   - `KeyType` represents the type of data being stored in the binary search tree. Your implementation should assume that
     - the `KeyType` class contains a key field and that the comparison operators for `KeyType` have been overloaded so that they return the result of comparisons between keys, and
     - the stream insertion operator (`<<`) has been overloaded for `KeyType`
   - The parameter of the `get` and `remove` methods is an object in which the internal key field is set to the value of the key for which to search. (Any other data fields in the object will be ignored.) The `get` method should return a pointer to the object in the hash table that contains the desired key.
   - Include suitable preconditions and postconditions in the comments before each method.
   - Your methods should throw appropriate exceptions when the parameters do not satisfy preconditions.
   - Include unit tests (using `assert` and the traversal methods) for each of your methods.

4. Implement a `Dictionary` ADT as a template class that inherits from your binary search tree template class. Your class should include public `get`, `insert`, `remove`, and `empty` methods.

5. Write a program that inserts information about movies (see the accompanying file) into a `Dictionary` object, and then allows for queries about the movies. Each movie has a title and a cast (a list of actor names). Your program should print the cast for any movie title entered.

In your submission, include all of your source files, and a single PDF containing your code and your answers to the questions, following the instructions in the previous projects. Be sure to indicate the names of both group members on all of your submitted files.